| **Ex No: 3b**  **Date: 22 Oct** | **Small Image Classification Using Convolutional Neural Network** |
| --- | --- |

**Objective:**The experiment's objective is to build, train, and evaluate ANN and CNN models for image classification using a CIFAR10 dataset from TensorFlow and Keras. It is a multi-class classification (10 classes) which aims to compare their performance by tuning hyperparameters, analysing accuracy, and drawing conclusions about which architecture is better suited for this task.

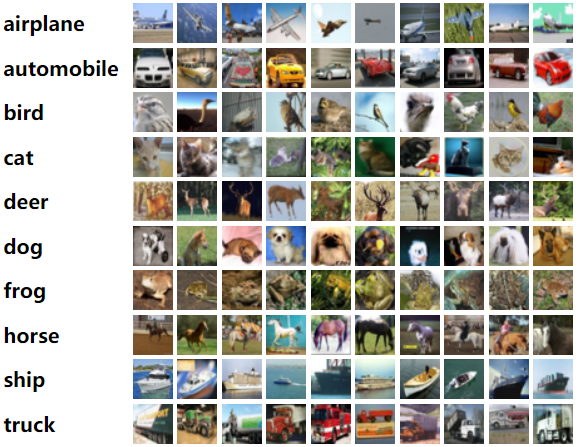
**Description:**

Artificial Neural Networks (ANNs) are machine learning models inspired by the human brain’s neural networks. They are composed of layers of neurons—input, hidden, and output—that process input data through nonlinear functions and learn patterns by adjusting weights during training. While ANNs work well for tabular and low-dimensional data, they may struggle with complex, high-dimensional data like images.

Convolutional Neural Networks (CNNs), a specialized type of ANN, are primarily designed for image and video analysis. They use convolutional layers to apply filters (kernels) to input data, detecting spatial hierarchies and features like edges, textures, and shapes. CNNs excel at image classification by capturing spatial relationships through convolutional operations and pooling layers.

Cifar-10 Dataset:

The CIFAR-10 dataset is a widely used benchmark dataset for image classification tasks in machine learning and computer vision. It consists of 60,000 color images, each 32x32 pixels, divided into 10 different classes, including objects like airplanes, cars, birds, cats, dogs, and more. The dataset is split into 50,000 training images and 10,000 test images, making it ideal for evaluating the performance of classification algorithms. Due to its relatively small size and diversity of classes, CIFAR-10 is popular for testing and developing deep learning models, particularly Convolutional Neural Networks (CNNs).

****

**Model Summary:**

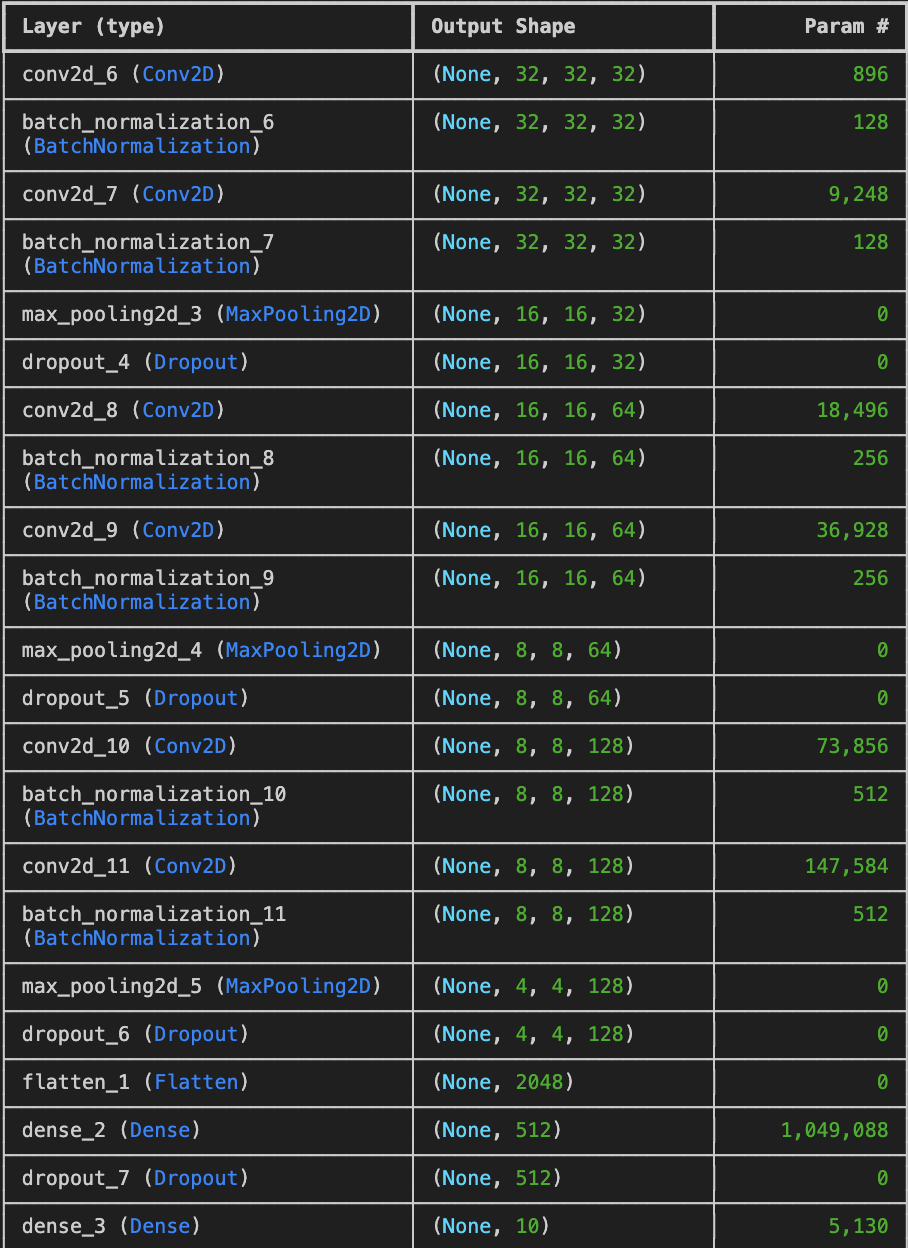
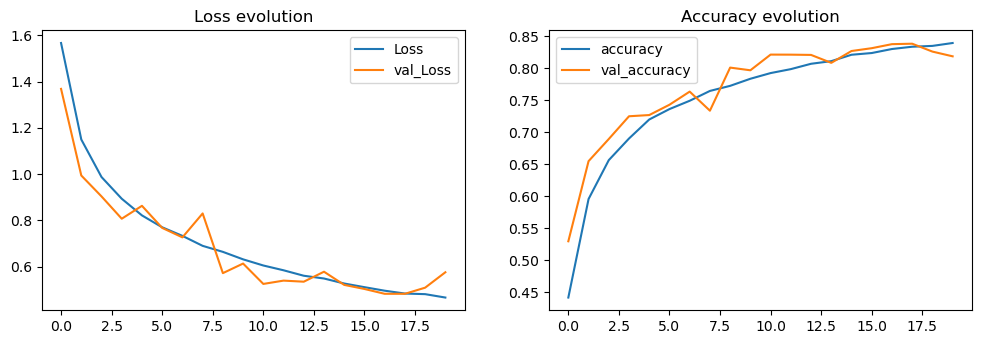
****

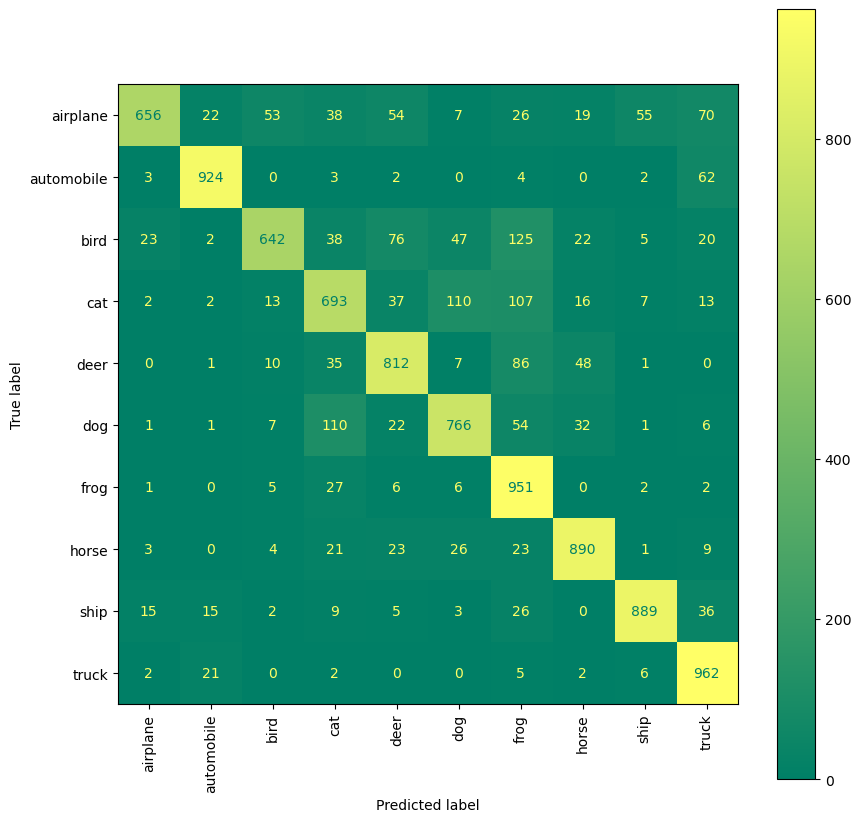
Figure 1: A sequential CNN model for Small Image classification

**Building the parts of the algorithm**

1. **Data Preprocessing:**
   1. The dataset is loaded and preprocessed into training and testing sets.
   2. Data augmentation is performed to enhance the variety of training images and reduce overfitting.
   3. The input data is normalized (e.g., pixel values scaled between 0 and 1).
   4. Class labels are converted into binary class matrices using one-hot encoding.
2. **Model Architecture:**
   1. **Input Layer:** The CNN model accepts images in their original shape (e.g., 32x32x3 for RGB images).
   2. **Convolutional Layers:** Filters (kernels) are applied to extract features like edges, textures, and patterns from the image data.
   3. **Activation Function:** ReLU is used after each convolutional layer to introduce non-linearity.
   4. **Pooling Layers:** Max pooling layers are applied to reduce the dimensionality of the feature maps and computation cost.
   5. **Flatten Layer:** The 2D feature maps are flattened into a 1D vector.
   6. **Fully Connected Layers:** Dense layers are used to integrate the extracted features for final decision-making.
   7. **Output Layer:** Softmax activation is used for multi-class classification, predicting probabilities for each class.
3. **Compilation:**
   1. The model is compiled using the categorical cross-entropy loss function.
   2. Optimizers used: SGD for ANN models and Adam for CNN models.
   3. Accuracy is set as the evaluation metric to track the performance of the model.
4. **Training:**
   1. The model is trained over multiple epochs, with real-time tracking of accuracy and loss during each epoch.
   2. Data augmentation (e.g., rotation, flipping, and zooming) is used to increase the robustness of the model by generating variations of the input images.
5. **Evaluation:**
   1. After training, the model’s performance is evaluated on the test set.
   2. The confusion matrix is plotted to provide a detailed view of the classification accuracy across different classes.
   3. Graphs for loss and accuracy over epochs are plotted to visualize model performance and trends.
6. **Graphs:**

****

**Fig 2: Loss and Accuracy Evolution**

****

**Fig 3: Confusion matrix**

**Inference:**

* The model achieved an accuracy of 81.2%, indicating that it correctly classified about 81% of the images in the test set.
* Data augmentation helped enhance the model’s ability to generalize by increasing the variety of training samples, which likely contributed to reducing overfitting.
* The confusion matrix provides deeper insights into class-wise performance, showing where the model performs well and where it struggles with misclassifications.

**Conclusion:**

For the project on **Small Image Classification Using Convolutional Neural Network**, the model achieved an accuracy of 81.2%, demonstrating its ability to effectively classify images across multiple categories. Data augmentation played a crucial role in improving generalization and reducing overfitting. However, while the model performed well overall, there are areas where further improvements can be made, such as fine-tuning hyperparameters or enhancing model complexity. The confusion matrix provides valuable insights into specific misclassifications, offering opportunities for targeted enhancements. Overall, the model is a solid foundation for small image classification tasks, with potential for refinement to achieve even better performance.

**GitHub Link:** [**https://github.com/Kashishvarmaa/DL-CS3232/blob/main/Lab\_3/cifar\_10.ipynb**](https://github.com/Kashishvarmaa/DL-CS3232/blob/main/Lab_3/cifar_10.ipynb)